Web Workers

Hasan Nawaz
2021-CS-52

1. Overview

This project entails a React application designed to fetch population data from the Data Faker object, offering two sorting mechanisms: one employing Web Workers for parallel processing and another without. The primary objective is to assess the performance disparities when sorting extensive datasets with and without the utilization of Web Workers within a React environment.

2. Performance Improvement Summary with Web Workers

The project serves to illustrate the implementation of Web Workers in executing sorting operations concurrently, potentially enhancing performance when managing substantial datasets. By delegating sorting tasks to distinct threads, Web Workers facilitate the maintenance of main thread responsiveness, thereby delivering a more seamless user experience.

3. Challenges Encountered and Resolutions

a. Web Worker Communication: Effective communication between the main thread and Web Workers necessitated meticulous handling. Leveraging the postMessage and onmessage events facilitated efficient data exchange.

b. Data Retrieval: Retrieving data from the Data USA API entailed asynchronous operations. Adhering to the async/await pattern facilitated the management of these operations, ensuring data availability prior to sorting.

4. References/Resources Utilized

a. Faker Library for Generating Synthetic Data (library)

b. MDN Web Docs - Web Workers: https://developer.mozilla.org/en-US/docs/Web/API/Web_Workers_API

c. React Documentation: https://reactjs.org/docs/getting-started.html

5. Conclusion

Employing Web Workers for computationally intensive tasks, such as sorting extensive datasets, holds the potential for enhancing performance in web applications. The concurrent processing capabilities of Web Workers enable efficient multitasking without compromising main thread responsiveness. However, it is imperative to assess the specific use case and task complexity to determine the most suitable approach. By locally executing the project and comparing sorting times with and without Web Workers, users can firsthand observe potential performance benefits, enabling informed decision-making based on application requirements.